

A three-dimensional domain decomposition method for large-scale DFT electronic structure calculations

Truong Vinh Truong Duy^{a,b,1,*}, Taisuke Ozaki^{a,1,**}

^a*Research Center for Simulation Science, Japan Advanced Institute of Science and Technology (JAIST), 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan*

^b*Institute for Solid State Physics, The University of Tokyo, Kashiwanoha 5-1-5, Kashiwa, Chiba 277-8581, Japan*

Abstract

With tens of petaflops supercomputers already in operation and exaflops machines expected to appear within the next 10 years, efficient parallel computational methods are required to take advantage of such extreme-scale machines. In this paper, we present a three-dimensional domain decomposition scheme for enabling large-scale electronic calculations based on density functional theory (DFT) on massively parallel computers. It is composed of two methods: (i) atom decomposition method and (ii) grid decomposition method. In the former, we develop a modified recursive bisection method based on inertia tensor moment to reorder the atoms along a principal axis so that atoms that are close in real space are also close on the axis to ensure data locality. The atoms are then divided into sub-domains depending on their projections onto the principal axis in a balanced way among the processes. In the latter, we define four data structures for the partitioning of grids that are carefully constructed to make data locality consistent with that of the clustered atoms for minimizing data communications between the processes. We also propose a decomposition method for solving the Poisson equation using three-dimensional FFT in Hartree potential calculation, which is shown to be better than a previously proposed parallelization method based on a two-dimensional decomposition in terms of communication efficiency. For

*Corresponding author. Tel.: +81 761 51 1987

**Principal corresponding author. Tel.: +81 761 51 1582.

Email addresses: duyvt@jaist.ac.jp (Truong Vinh Truong Duy),
t-ozaki@jaist.ac.jp (Taisuke Ozaki)

¹These authors contributed equally to this work.

evaluation, we perform benchmark calculations with our open-source DFT code, OpenMX, paying particular attention to the $O(N)$ Krylov subspace method. The results show that our scheme exhibits good strong and weak scaling properties, with the parallel efficiency at 131,072 cores being 67.7% compared to the baseline of 16,384 cores with 131,072 diamond atoms on the K computer.

Keywords: First principles calculations; Linear scaling method; Krylov subspace method; Domain decomposition; Modified recursive bisection; Inertia moment tensor; FFT grid decomposition

1. Introduction

Density functional theory (DFT) [1, 2, 3] is widely regarded as one of the most powerful and popular methods available in computational materials science simulations. It has been applied to various systems of interest, from physics, chemistry, and materials science to biology, for exploring a wide range of material and biological properties, such as structural and optical properties, electric transport, chemical reactions, etc. Although the Kohn-Sham (KS) method offers a feasible approach to performing the calculation of the ground state properties of N -body systems, its scaling property of $O(N^3)$ has posed great challenges in large-scale calculations. There has been considerable effort to realize large-scale DFT calculations that can be roughly classified into two major approaches: first, improve and implement highly efficient parallelization of conventional methods, and second, develop better-than-cubic scaling methods, ideally linear scaling methods.

In the first approach, real-space methods are well-established and significant progress in developing practical real-space methods and implementations has been observed. Recently, a real-space DFT code has achieved an unprecedented performance on the K computer [4]. Real-space implementations of DFT applying finite difference methods [5] and finite element methods [6, 7, 8, 9] have also been extensively developed. On the other hand, in the second approach, a number of linear scaling methods have been proposed. Several survey reports on linear scaling methods can be found in literature [10, 11, 12, 13, 14]. A popular and straightforward approach to linear scaling is the divide and conquer (DC) method [15, 16, 17]. Recursion method proposed by Ozaki et al. in [18, 19, 20] is another approach. Other linear scaling methods include the early work presented in [21], methods based on

Wannier functions [22, 23, 24, 25] and density matrix [26, 27, 28, 29], and methods for tight-binding systems [30, 31] and for metallic and non-metallic systems [32, 33].

Our main concern in this paper is the development of an efficient three-dimensional domain decomposition method that is applicable to both conventional methods and linear scaling methods to enable large-scale electronic calculations based on DFT on massively parallel computers. In fact, our linear scaling DFT method based on Krylov subspace method has been developed and introduced in [34]. Even though there exist a number of parallel implementations of linear scaling DFT [35, 36, 37] and domain decomposition methods based on the space-filling curve technique [38, 39], these methods may suffer from difficulty in applying to anisotropic structures. Considering the fact that we are now in the petaflops era with several supercomputers having a peak performance of tens of petaflops and exaflops machines are expected to appear within the next 10 years with millions of cores, generally applicable and efficient methods are required to take advantage of such extreme-scale machines and solve a wide variety of problems. In parallelization of linear scaling methods, as our implementation is also based on atomic orbitals [40] and their linear combination [41, 42], a known problem arises due to the use of localized atomic basis functions with different spatial cut-off radius for different elements. The irregular sparse structure of the resultant matrices make their parallel multiplication implementation become more complicated, since they require careful mapping of the sparse matrix elements to processing cores, unlike finite element methods where the matrices are regular. In addition, the domain decomposition method should hold the locality of clustered atoms in real space and assign nearby atoms to processors to minimize inter-node communications. Also, it should be applicable to any distribution patterns of atoms in real space.

To address the issues, in this paper we propose a domain decomposition scheme, which is actually composed of two methods: one for decomposing the atoms and the other for partitioning the grids among the processes. In the former, we develop a modified recursive bisection method based on inertia tensor moment to reorder the atoms along a principal axis so that atoms that are close in real space are also close on the axis. Depending on their positions on the axis, the atoms are then divided into sub-domains in a balanced way with data locality conserved. The method should be able to work well with any number of atoms and processes. Meanwhile, in the latter, we introduce four data structures for the grid partitioning

that are carefully constructed to make data locality consistent with that of the clustered atoms, resulting in the minimization of data communications between the processes. We furthermore propose a decomposition method for solving the Poisson equation using three-dimensional (3D) FFT in Hartree potential calculation, which is proven to be more communication efficient than one- and two-dimensional decomposition methods. We also let the processes perform on-the-fly communications to reduce the memory usage.

The remainder of the paper is organized as follows. Section 2 describes briefly the research background, including the KS equation and our linear scaling Krylov subspace method. The domain decomposition methods for atoms and grids are presented in Section 3. Section 4 details implementation issues and the calculation flowchart. Section 5 analyzes benchmark results with a focus on linear scaling, and strong and weak scaling properties. Finally, we conclude our study in Section 6.

2. Background

For the sake of completeness and self-explanation, we briefly introduce the background of DFT by way of the well-known KS equation, and give an overview of our linear scaling Krylov subspace method, while highlighting data structures related to our parallelization scheme.

2.1. Kohn-Sham equation

In principle, DFT is a modeling method to determine the electronic ground state of N -body systems of atoms and molecules based on functionals of electron density. Although DFT may have its root date back to the Thomas-Fermi model and Slater’s $X\alpha$ method derived from a simplification of the Hartree-Fock method [43, 44], its theoretical foundation was only established firmly by the Hohenberg-Kohn theorems introduced in the 1960s [1]. The resulting KS equation is typically represented in the form of an eigenvalue equation as below [45, 46].

$$\hat{H}_{\text{KS}}\phi_{\nu}(\mathbf{r}) = \varepsilon_{\nu}\phi_{\nu}(\mathbf{r}), \quad (1)$$

where \hat{H}_{KS} is the KS Hamiltonian, and ε_{ν} is the orbital energy of the corresponding KS orbital $\phi_{\nu}(\mathbf{r})$. Consequently, the density $n(\mathbf{r})$ of the system is defined as:

$$n(\mathbf{r}) = 2 \sum_{i=1}^{\text{occ.}} \phi_{\nu}^{*}(\mathbf{r})\phi_{\nu}(\mathbf{r}), \quad (2)$$

where the factor of 2 is due to spin multiplicity, and ν runs for all the occupied states. Although our discussion is limited within non-spin polarization, the generalization to spin polarization is straightforward.

In our implementation, we utilize atomic orbital basis functions, written as the multiplication of a spherical harmonic Y_l^m and a radial function R :

$$\phi_\nu(\mathbf{r}) = Y_l^m(\hat{\mathbf{r}})R(r). \quad (3)$$

The basis functions have been carefully constructed and optimized for convergence and reduction of computational effort with high accuracy, as well as memory usage [42]. Most notably, the use of atomic-like orbitals perfectly matches the idea of linear scaling methods. They are strictly confined within a sphere and stored in a special structure described later in Section 3.2.2. In the same section, another structure for storing all the grid points inside a parallelepipedon defined by the basis functions of all atoms distributed to the same process in the parallel implementation will also be presented.

The Hamiltonian is expressed as:

$$\hat{H}_{\text{KS}} = -\frac{1}{2}\nabla^2 + v_{\text{eff}}(\mathbf{r}), \quad (4)$$

where $v_{\text{eff}}(\mathbf{r})$ is the Kohn-Sham effective potential, given by:

$$v_{\text{eff}}(\mathbf{r}) = v_{\text{ext}}(\mathbf{r}) + v_{\text{Hartree}}(\mathbf{r}) + \frac{\delta E_{\text{xc}}[n]}{\delta n(\mathbf{r})}. \quad (5)$$

Here, the first term, $v_{\text{ext}}(\mathbf{r})$, is the external potential due to the interaction between electrons and nuclei and other external fields. The second term $v_{\text{Hartree}}(\mathbf{r})$ is the classical Hartree (Coulomb) interaction of the electron density interacting with itself, given by

$$v_{\text{Hartree}}(\mathbf{r}) = \int \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}'. \quad (6)$$

In our implementation, the Hartree potential is found by solving the Poisson equation using FFT. In parallel implementations, the grid points must be decomposed to the processes in a proper fashion to minimize the communication amounts incurred during the calculation. We will introduce our grid decomposition method and compare with other methods in terms of communication efficiency in Section 3.2.

The last term in Eq. (5) is the exchange-correlation potential, where E_{xc} is the corresponding exchange-correlation energy that is the unknown in this KS approach. A number of methods have been proposed to yield approximations to this exchange-correlation energy, for instance, local density approximation (LDA) and generalized gradient approximation (GGA) [45, 46]. We will construct a particular structure for the calculation of this potential in Section 3.2.2.

The KS equation turns out to be in the form of one-body equations under the effective potential. Combined with the methods for approximating E_{xc} , the equation can be solved self-consistently with a resulting ground state density. Then the corresponding energy can be determined:

$$E_{\text{KS}}[n] = T_s + \int n(\mathbf{r})v_{\text{ext}}(\mathbf{r})d\mathbf{r} + E_{\text{Hartree}}[n] + E_{\text{xc}}[n], \quad (7)$$

where T_s is the KS kinetic energy of non-interacting electrons, also expressed in terms of the KS orbitals:

$$T_s = - \sum_{\nu=1}^{\text{occ.}} \int \phi_{\nu}^* \nabla^2 \phi_{\nu} d\mathbf{r}. \quad (8)$$

2.2. Linear scaling Krylov subspace method

The basic idea behind the Krylov subspace method [34] is to combine the DC method and the recursion method based on the Green's function. The DC method has been proven good for covalent systems but not for metals due to the difficulty of direct diagonalization for large truncated clusters, while the recursion method is known for its numerical instability in SCF calculations. The three methods are founded on the same fact that the charge density n can be given by summing up the local charge density n_i of each site, which is actually evaluated with the density matrix ρ :

$$n(\mathbf{r}) = 2 \sum_{i\alpha, j\beta} \phi_{i\alpha}(\mathbf{r}) \phi_{j\beta}(\mathbf{r}) \rho_{i\alpha, j\beta} = 2 \sum_i \left(\sum_{\alpha, j\beta} \phi_{i\alpha}(\mathbf{r}) \phi_{j\beta}(\mathbf{r}) \rho_{i\alpha, j\beta} \right) = 2 \sum_i n_i(\mathbf{r}), \quad (9)$$

where i is the site index, and α is the orbital index of the basis functions.

The density matrix ρ is in turn calculated by the one-particle Green's function:

$$\rho_{i\alpha, j\beta} = -\frac{1}{\pi} \text{Im} \int G_{i\alpha, j\beta}(E + i0^+) f\left(\frac{E - \mu}{k_B T}\right) dE, \quad (10)$$

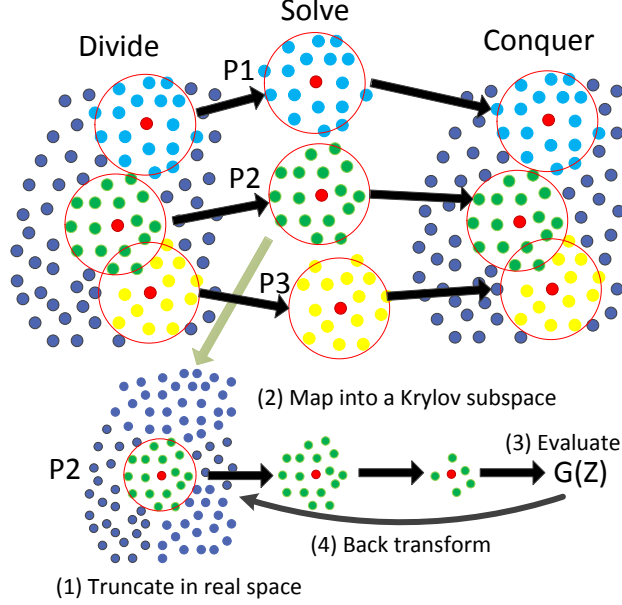


Figure 1: Linear scaling Krylov subspace method.

where $f(x) \equiv 1/[1 + \exp(x)]$ is the Fermi function, and 0^+ is a positive infinitesimal.

The problem now turns to how to evaluate the Green's function in linear scaling time, and hence each method has its own approach. The Krylov subspace method can be seen as a DC method defined in a Krylov subspace for reducing the size of the truncated clusters, as the Krylov subspace is much smaller than the original vector space. In this method, the total Green's function can be approximated by the sum of the local Green's functions associated with the central atom in each truncated cluster.

Figure 1 demonstrates how the Krylov subspace method works in practice. First, in the Divide step, the original system is decomposed into smaller truncated clusters using a physical truncation scheme, described later in Section 4. The truncated clusters are assigned to different processes for processing in parallel. Then, in the Solve step, the Krylov subspace for each truncated cluster is generated independently on each process without any communications. After that the Green's function associated with each truncated clusters is evaluated, and the back transform is performed to get the Green's functions represented by the original basis functions for calculating the density

matrix ρ which is then found by Eq. (10), followed by the local charge density n_i by Eq. (9). Finally, in the Conquer step, the total charge density n is given by combining all the local charge densities n_i (Eq. (9)). The density is treated by regular mesh, and we will therefore introduce the grid decomposition method with four data structures for storing data on the grid points in Section 3.2. The Krylov subspace method has been proven to be efficient, robust, could converge rapidly for both metals and insulators, and has been recently applied to a large-scale molecular dynamics simulation for electrochemical systems [47] and structural prediction of precipitates of a carbide in bcc iron [48]. Interested readers are referred to [34] for details.

3. Domain decomposition methods

Domain decomposition should be performed in three dimensions, especially for large-scale calculations with a large number of processes, considering the scaling properties of communication amount and memory usage. One- and two-dimensional decomposition methods are only appropriate for up to a certain number of processes, and when that number exceeds the limit of one or two dimensions, the amount of communication and memory usage will stay unchanged. In contrast, three-dimensional methods can realize much larger scales of calculations, as they are perfectly suited to the three-dimensional nature of systems' structure. In this section, we present our three-dimensional domain decomposition scheme, consisting of two methods: one for the atom level and the other for the grid level in three-dimensional space.

3.1. Three-dimensional atom decomposition method

3.1.1. Requirements

In parallel implementations, atom decomposition among the processes is a very important first task that has a considerable impact on the overall performance. We consider a 3D domain decomposition method for atoms in a system, which should hold the locality of clustered atoms in real space with the following requirements.

- **Approximately the same computational amount:** Each sub-domain should have approximately the same computational amount. This requirement is obvious for ensuring a good load balance among the processes. As atoms of different elements have different processing time, each atom should have a corresponding weight reflecting the

amount of computation. The only exception is the case of the first MD step or geometry optimization step, where all the atoms have the same weight of 1.

- **Locality:** Nearby atoms should be grouped together and assigned to the same process. This will reduce the amount of communication and assists the development of linear scaling methods, which are also essentially based on the idea of locality.
- **Applicable to any number of atoms and processes:** Realistic systems can have any number of atoms, and can be solved by any number of processing cores available to maximize resource usage. Hence, in order to treat them effectively, the method should be able to work with any number of atoms and processes.
- **Applicable to any distribution pattern of atoms:** Some specific systems have most atoms distributed around the center of mass, while some have atoms spread uniformly in the space, and others may have a majority of atoms lie along just one main dimension. Such irregular distributions of atoms should be considered in the method.
- **A well-defined algorithm:** To be clear and easy to follow.

Figure 2 exhibits an expected domain decomposition result for a system of 26 atoms, in 2D for the sake of simplicity, where nearby atoms are nearly equally allocated into four sub-domains. Given the requirements, a proper 3D domain decomposition can be performed based on two ideas: modified recursive bisection and inertia moment tensor, which are presented subsequently.

3.1.2. Modified recursive bisection method

The modified recursive bisection method is a useful tool to decompose the system with any number of processes and atoms, unlike the conventional recursive bisection method that is restricted to numbers equal to a power of 2 [49]. The method involves constructing a binary tree, and each tree node has a number representing the number of processes treating the domain under that node. Leaf nodes have a number of processes valued at 1, and the number of leaf nodes is equal to the number of processes. In practice, the method works as follows.

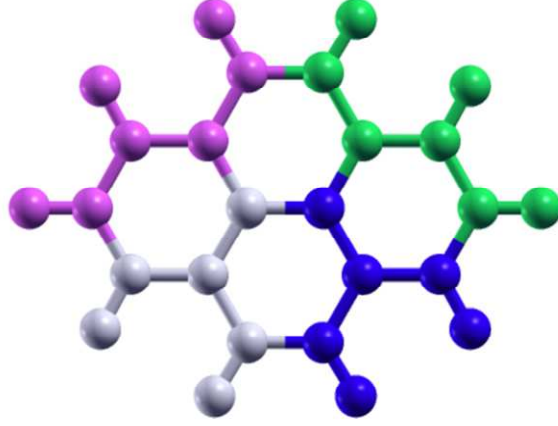
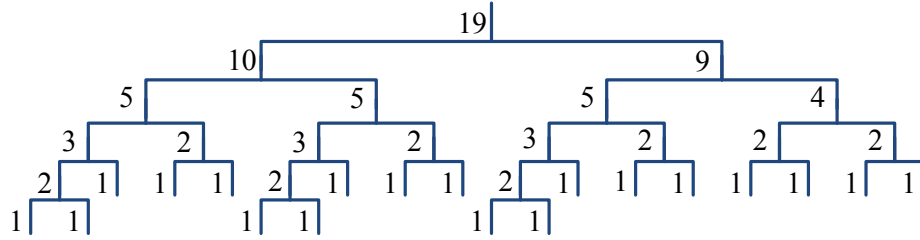


Figure 2: An example of domain decomposition for atoms with each color representing a sub-domain.

First, the original domain is divided into two sub-domains, each having approximately the same number of processes, which is about half of that of the original domain. Each sub-domain is then further divided into two, again with each having nearly the same number of processes. The division is repeated until the number of processes becomes 1. Once the division procedure has been completed, there are as many leaf nodes as processes.

Figure 3 shows an example of the method where the number of processes is 19. It is also important to note that in the conventional recursive bisection method limited to the number of processes equal to a power of 2, the bisection is made so that equal numbers can be assigned to each sub-domain. However, the modified method bisects with weights as shown in the figure.



3.1.3. Inertia tensor moment

If the modified recursive bisection method meets the requirement to work with any number of processes and atoms, the use of an inertia moment tensor for the bisection in the method ensures the locality of the domain decomposition. The inertia moment tensor is applied to calculate a principal axis for each sub-domain during the decomposition process described above. Then atoms in the corresponding sub-domain are reordered from three dimensions to one dimension by projecting them onto the principal axis. This reordering will also make it much easier to divide the atoms into two sub-domains to fit the structure of the binary tree. The inertia tensor moment approach is expected to work well, as atoms that are close in real space are also close on the principal axis yielded by the inertia tensor. Figure 4 illustrates the idea of 3D-to-1D reordering of atoms.

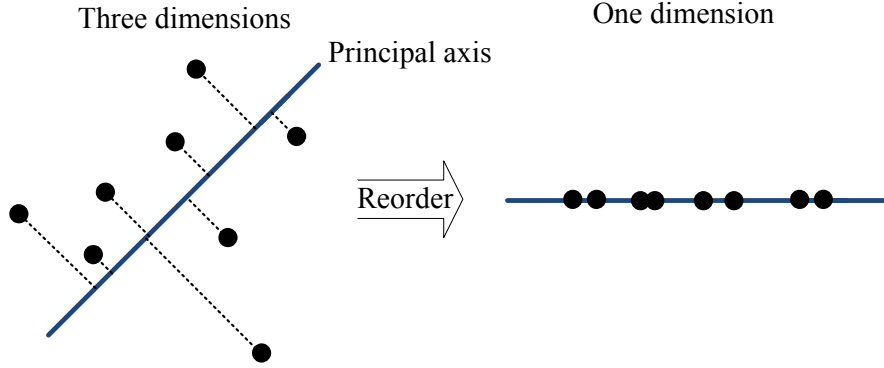


Figure 4: Reordering of atoms by a principal axis found by the inertia tensor moment.

Mathematically, the equation of the principal axis can be written as:

$$\begin{aligned} x &= C_x + a_x t \\ y &= C_y + a_y t \\ z &= C_z + a_z t, \end{aligned} \tag{11}$$

where (C_x, C_y, C_z) is the center of mass of the system, defined as the average of the atoms' positions, x_i , weighted by their weights, w_i , with the total number of atoms, N_a :

$$\begin{aligned} C_x &= \frac{1}{N_a} \sum_{i=1}^{N_a} x_i w_i \\ C_y &= \frac{1}{N_a} \sum_{i=1}^{N_a} y_i w_i \\ C_z &= \frac{1}{N_a} \sum_{i=1}^{N_a} z_i w_i. \end{aligned} \tag{12}$$

Projection t_i of an atom i on the principal axis is given by:

$$t_i = a_x(x_i - C_x) + a_y(y_i - C_y) + a_z(z_i - C_z). \quad (13)$$

We then define a function F as:

$$F = \sum_i w_i t_i^2 - \lambda(|a|^2 - 1). \quad (14)$$

The function F is now the target for maximization to find the possible longest principal axis onto which all atoms can be projected. The derivatives of F over the coordinate axes:

$$\frac{\partial F}{\partial a_x} = \frac{\partial F}{\partial a_y} = \frac{\partial F}{\partial a_z} = 0 \quad (15)$$

lead to an eigenvalue equation:

$$T \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \lambda \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}. \quad (16)$$

T is the tensor of inertia about the center of mass with respect to the xyz axes, written in a matrix form as:

$$T = \begin{pmatrix} \sum_i w_i(Y_i^2 + Z_i^2) & -\sum_i w_i X_i Y_i & -\sum_i w_i X_i Z_i \\ -\sum_i w_i Y_i X_i & \sum_i w_i(X_i^2 + Z_i^2) & -\sum_i w_i Y_i Z_i \\ -\sum_i w_i Z_i X_i & -\sum_i w_i Z_i Y_i & \sum_i w_i(X_i^2 + Y_i^2) \end{pmatrix}, \quad (17)$$

where

$$\begin{aligned} X_i &= x_i - C_x \\ Y_i &= y_i - C_y \\ Z_i &= z_i - C_z. \end{aligned} \quad (18)$$

Finally, the principal axis is found by solving the eigenvalue problem with the inertia tensor. After solving the eigenvalue problem, we obtain three eigenvalues and the corresponding eigenvectors. Among the three states, we choose the state that maximizes the function F in Eq. (14). Also, the derivations imply that there are many ways to properly decompose a system by changing the definition of the function F . In our method, it is expressed as a maximization problem to find the correspondingly longest principal axis so that all atoms can be projected onto it.

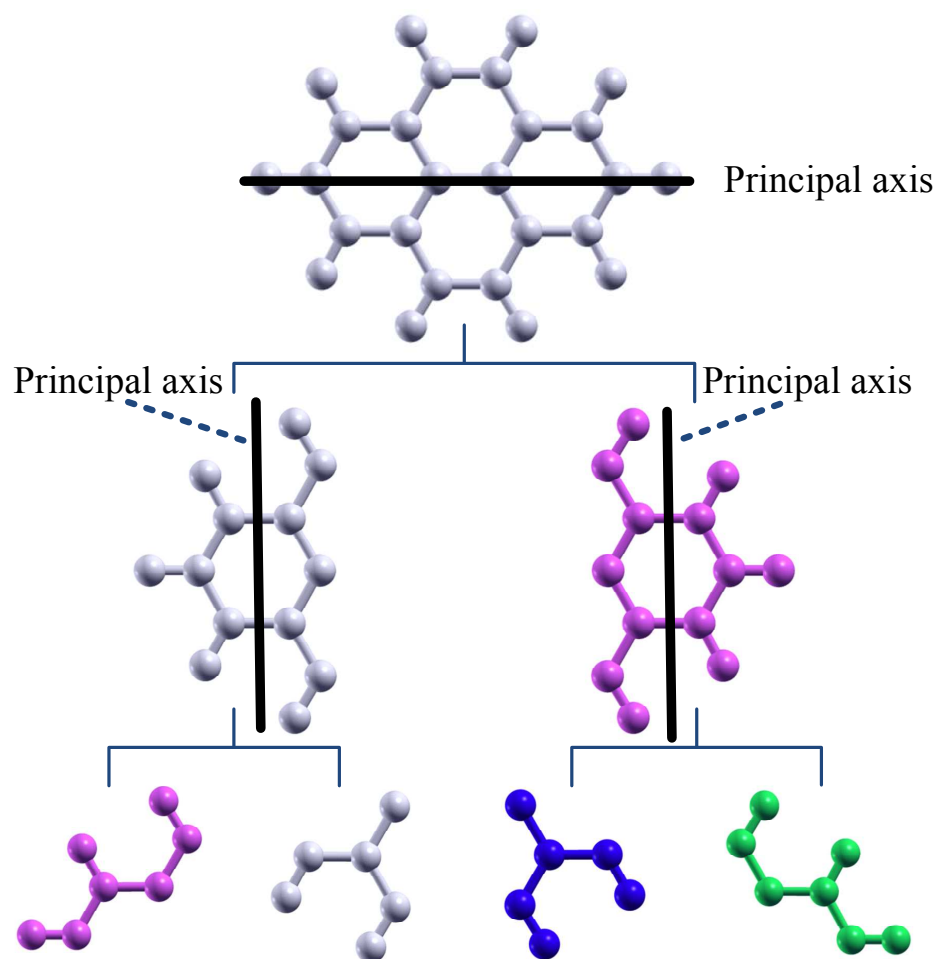


Figure 5: Atom decomposition with the modified recursive bisection method and the principal axis.

Figure 5 exemplifies the operation of our atom domain decomposition scheme based on the modified recursive bisection method and the principal axis. The binary tree is constructed with the root representing the original domain consisting of 26 atoms. The principal axis of the original domain is then found, based on which the domain is divided into two sub-domains with each having 13 atoms. The sub-domains are assigned to the two child nodes of the tree and the process of finding the principal axis and dividing the domain is repeated for each sub-domain. The domain on the left child node is divided into two sub-domains with one having 7 atoms and the other having 6 atoms, and so is the domain on the right child node. The decomposition is performed recursively until there are as many sub-domains as processes.

3.1.4. Load balancing

As different atoms may have different processing time depending on their properties, each atom is associated with a weight for load balancing. As a result, both the number of atoms and weights are considered in the decomposition method in an effort to distribute the amount of computation equally among the processes. The weight of an atom i is defined as:

$$w_i = \frac{Etime_i}{Mtime}, \quad (19)$$

where $Etime_i$ is the processing time of the atom i in the previous MD step, and $Mtime$ is the longest processing time of all atoms.

It should be noted that the weight definition has an important impact upon reordering the atoms. The weight defined in Eq. (19) tends to position light atoms at around the center, while heavy atoms tend to move away from the center. As the bisection for dividing the domain is usually found near the center, the decomposition is indeed more precise when light atoms are also positioned around it. Therefore, our method based on the inertia tensor moment fulfils the requirement for load balancing and locality.

3.1.5. Algorithm for atom decomposition

Figure 6 outlines the algorithm for the atom decomposition method with a combination of the modified recursive bisection method and the inertia tensor. It starts with the root node of the binary tree representing the original domain as the current node. If the number of processes is 1 or the number of atoms is 0, the current node is the leaf node and the algorithm will stop. Otherwise, it will find the center of mass with weights of the current

domain (Eq. (12)), and compute the inertia tensor based on Eq. (17). After that, the inertia tensor is diagonalized to obtain the principle axis. Now the projection t_i of each atom onto the principal axis can be determined using Eq. (13). They are then sorted to find the bisection for dividing the current node into two child nodes based on Eq. (19) where the sum of weight is nearly equivalent to each other. The algorithm repeats with each child node until reaching the leaf node, where there is only one assigned process or no atom available.

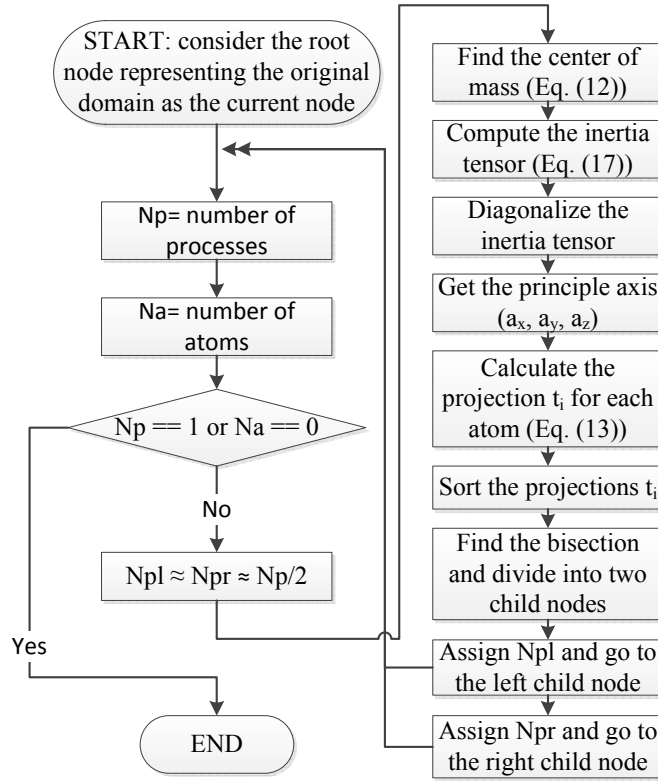


Figure 6: The algorithm for decomposing atoms.

3.2. Three-dimensional grid decomposition method

As the regular mesh technique is used in our implementation [50], after the atoms have been distributed to the processes, the grid points belonging to their basis functions must also be allocated to the corresponding ones. Once the density matrix ρ has been calculated by Eq. (10), the charge density can

be found by Eq. (9) with the basis functions using regular mesh. As the Hartree potential needs to be calculated by FFT using the whole system, the allocation of the grids to the processes should be carried out effectively so that the communication amount among them is as small as possible.

3.2.1. Requirements

In parallel with the atom decomposition method, a 3D domain decomposition method for grids is also developed with the following requirements.

- **Approximately the same number of grid points:** Similar to the case of atom decomposition, processes should be assigned nearly the same number of grid points so that they can perform approximately the same computational amount.
- **Almost linear scaling in memory usage:** Memory usage is always a source of concern in DFT-based calculations, as a large amount of memory is demanded for storing a huge number of grid points. As such, the amount of memory required for serial calculations should be only linear to the system size, and should remain a constant for parallel calculations as a function of the numbers of atoms and processes when the same number of atoms is allocated to each process.
- **Locality:** Grid points within the sphere of a basis function (Eq. (3)) belonging to an atom that is allocated to a process should be assigned to the same process with the atom. The locality should also be held when distributing the grids to the processes. This will have two-fold benefits: minimize the amount of communication and the amount of memory necessary for storing data on the grid points overlapping with those of the basis functions for all atoms in one process.

3.2.2. Data structures for MPI parallelization

To comply with the requirements, here we define four data structures to make the data locality consistent with that of the clustered atoms for minimizing the amount of communication between the processes, which are shown in Fig. 7.

- **Structure A:** This structure stores all the grid points inside the sphere of a basis function.

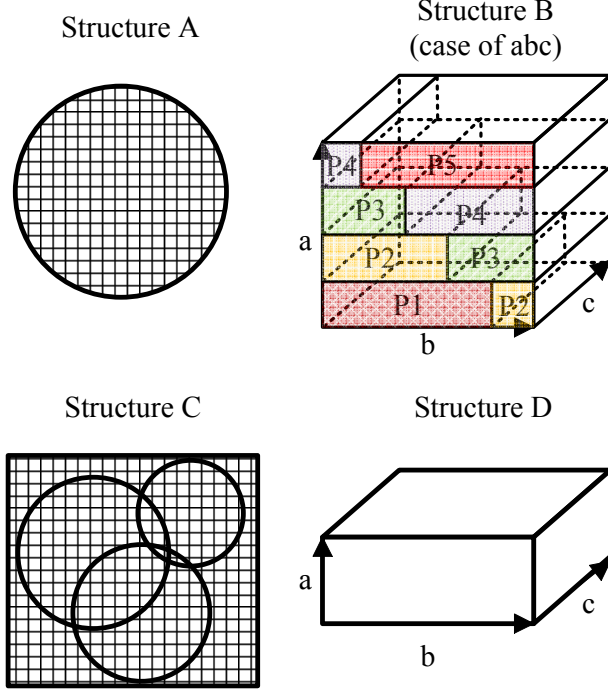


Figure 7: Data structures for MPI parallelization.

- Structure B:** This structure stores all the grid points allocated to a specific process. Here we propose a two-dimensional decomposition method for parallel 3D FFT, where the distribution of the grid points is carried out in a two-dimensional grid defined by the first two dimensions. Therefore, the structure B may exist in different forms by changing the order of distribution. For example, in case of **abc** distribution, the grid points on the **ab**-plane with the **c** axis are divided over the processes in ascending order of their **a** and **b** coordinates so that each process has approximately the same number of grid points on the **ab**-plane. The grid points extending along the **c** direction that have the same **a** and **b** coordinates are also assigned to the same process. Assume that the numbers of grids of the **a**-, **b**-, and **c**-axes are N_a , N_b , and N_c , respectively, the number of processes is N_p , and $myid$ is the process's ID. In our method, the process with $myid$ will be assigned the grid points from the coordinate of $(a_s, b_s, 0)$, where $a_s N_b + b_s = \left\lfloor \frac{myid \times N_a N_b + N_p - 1}{N_p} \right\rfloor$, to the coordinate of $(a_e, b_e, N_c - 1)$,

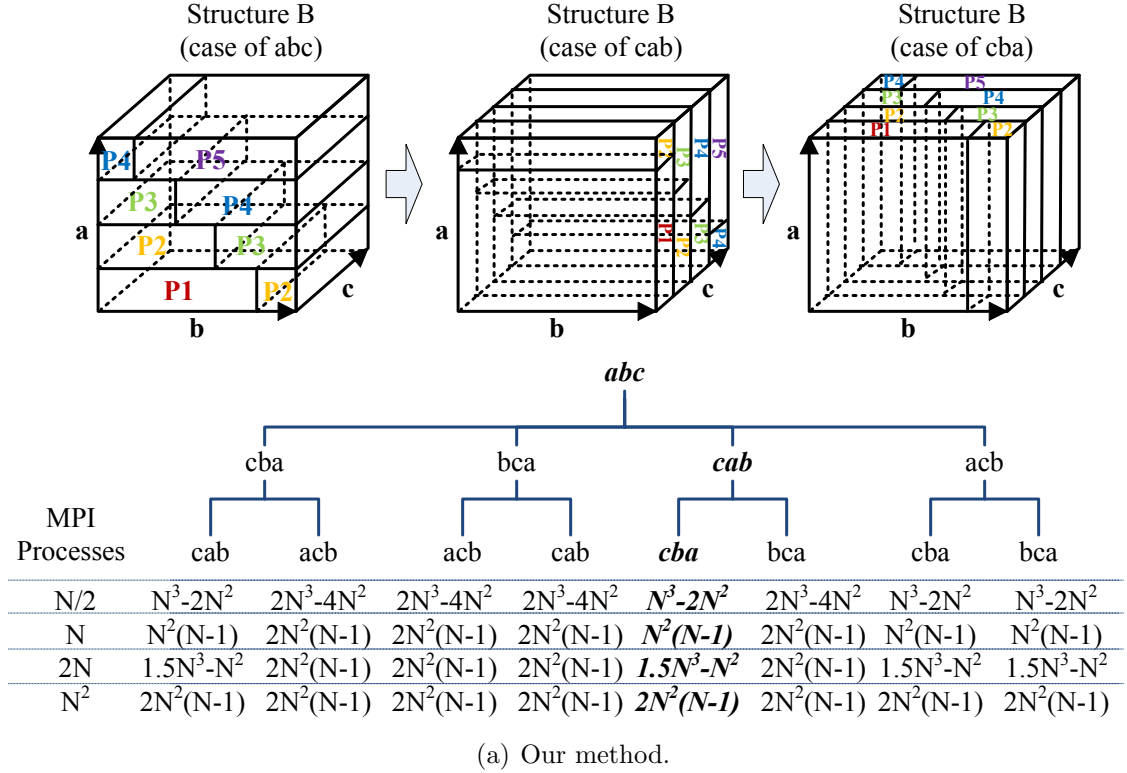
where $a_e N_b + b_e = \left\lfloor \frac{(myid+1) \times N_a N_b + N_p - 1}{N_p} - 1 \right\rfloor$, in ascending order of their **a**, **b**, and **c** coordinates. As a result, our decomposition method can be viewed as a hybrid method between one-dimensional decomposition and two-dimensional decomposition, since it is the same as the former for up to a certain number of processes, and is gradually akin to the latter for larger numbers of processes. Detailed analysis on the communication amount will be given later in 3.2.3.

- **Structure C:** The structure stores all the grid points inside a parallelepipedon defined by the basis functions of all atoms allocated to one process, paying attention to the cutoff radius of the basis functions.
- **Structure D:** This structure can be seen as an extension of the structure B(**abc**) because it is created by adding some buffer regions around the structure B. Strictly speaking, this structure is only necessary for GGA [51] as it requires neighboring grid points for calculating the gradient of the charge density by a finite difference method. Meanwhile, LDA [52] can be performed with the structure B without having to refer to the structure D.

The introduction of the data structures makes it more straightforward in the calculations. For example, one can easily find a specific basis function by referring to the structure A. As can be seen subsequently, the structure B is related to a two-dimensional decomposition method for FFT that is more efficient than previously proposed methods in terms of communication amount.

3.2.3. MPI communication analysis in the domain decomposition method for 3D FFT

In our two-dimensional decomposition method for parallel 3D FFT, as different distributions of the structure B are involved in the calculations in different orders, the order of reference may have an impact on communication amount. Figure 8 shows an analysis on the amount of communication to transpose the structure B from one to another depending on the number of processes, in comparison with a traditional one-dimensional decomposition method and a two-dimensional decomposition method [53]. Interestingly, the amount of communication is grouped into only two patterns of communication in our method, and there is one pattern actually better than the other.



Method	Number of MPI Processes			
	N/2	N	2N	N ²
1D Method	N ³ -2N ²	N ² (N-1)	Unapplicable	Unapplicable
2D Method [53]	2N ³ -N(2N) ^{3/2}	2N ³ -2(N) ^{5/2}	2N ³ -4N(N/2) ^{3/2}	2N ² (N-1)

(b) One- and two-dimensional methods.

Figure 8: MPI communication analysis in the parallelization of 3D FFT for (a) our method and (b) other one-dimensional method and two-dimensional method [53] with $N \times N \times N$ grid points.

In general, we can choose any of the four orders leading to the pattern with the smaller amount of communication, for instance $abc \rightarrow cab \rightarrow cba$, or $abc \rightarrow cba \rightarrow cab$. However, we choose the order of $abc \rightarrow cab \rightarrow cba$ to keep consistency with our current implementation of other functionalities such as non-equilibrium Green's function (NEGF) method.

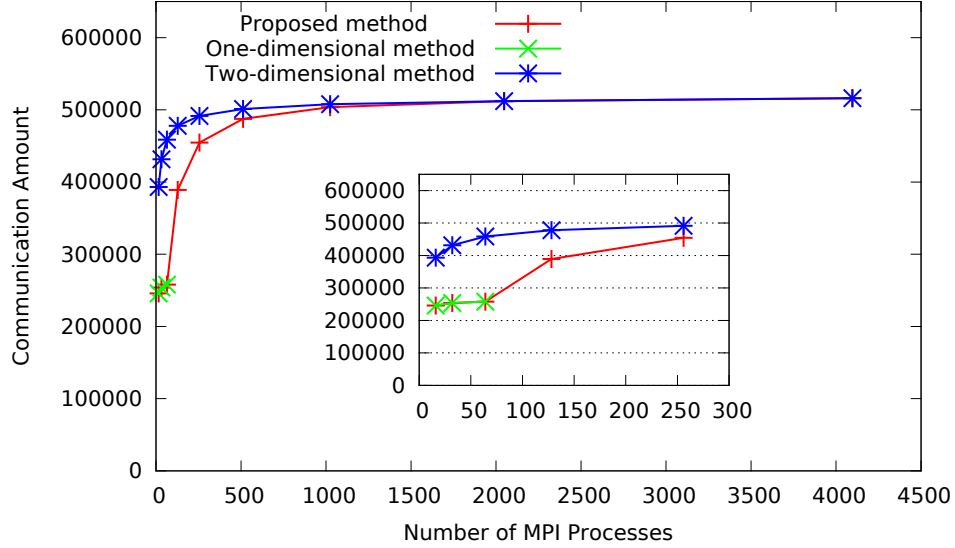


Figure 9: Comparison of communication amount in the parallelization of 3D FFT in case of $64 \times 64 \times 64$ grid points ($N = 64$). The inset enlarges the left part of the main graph.

Figure 9 shows a comparison among our proposed method and the two other methods in terms of communication amount in case of $64 \times 64 \times 64$ grid points. Certainly, similar results can be produced for smaller and larger numbers of grid points. In this case, the one-dimensional method is able to work only along one dimension and is limited to 64 processes, while the two-dimensional method always decomposes the domain in two dimensions, even for fewer than 64 processes. In contrast, our method offers a combination of these methods, as it partitions only along one dimension when the number of processes is up to 64 on condition that it is a divisor of 64, and decomposes in two dimensions while still starting from one dimension for larger process numbers. In other words, our method is based on a row-wise decomposition, as against a square decomposition approach taken in [53]. In doing so, most of communication amount is incurred when transferring from abc to cab , and a majority of data are reused when transferring from cab to cba with just a

small amount of communication. As a result, up to 64 processes, our method works in the same fashion as the one-dimensional method provided that 64 is a multiple of the number of processes, and is about 60.0% to 77.8% better than the two-dimensional method with $64 \times 64 \times 64$ grid points. Beyond this point, the one-dimensional method is no longer applicable, while the two other methods can operate until reaching the limit of $64 \times 64 = 4096$ processes. The difference in performance gradually decreases, however, and eventually becomes 0 from 2048 processes.

3.2.4. Data structures and calculation flow

The relationship between the data structures and the calculation flow is depicted in Fig. 10. The calculation of charge density (Eq. (9)) is performed with the structures A, B, and C in order of appearance. The Hartree potential (Eq. (6)) is calculated by solving the Poisson equation using FFT for the whole system with different distributions of the structure B in the order of $\mathbf{abc} \rightarrow \mathbf{cab} \rightarrow \mathbf{cba} \rightarrow \mathbf{cab} \rightarrow \mathbf{abc}$. The exchange-correlation potential (the last term in Eq. (5)) is determined from either the structure C or the structure D depending on the approximation method in use. The structure B is also utilized for performing the charge mixing ($\mathbf{cab} \rightarrow \mathbf{cba} \rightarrow \mathbf{abc}$), and for calculating the total energy (Eq. (7)). It should be noted that communication is required when the calculations move from one structure to another.

4. Implementation

Figure 11 illustrates the implementation flowchart in terms of the computational flow that consists of three key steps: atom decomposition, grid decomposition, and $O(N)$ calculation. The first two steps are common, and applicable to other schemes such as conventional DFT calculations, NEGF method, etc., while the last step is specially for our linear scaling Krylov subspace method.

- Step 1 **Atom decomposition.** The atoms are allocated to the processes by the proposed atom decomposition method based on a combination of the modified recursive bisection method and inertia tensor. The algorithm describing its operation is detailed in section 3.1.5.
- Step 2 **Grid decomposition.** The grid points are decomposed using our grid decomposition method. It constructs the structures A, B, C, and

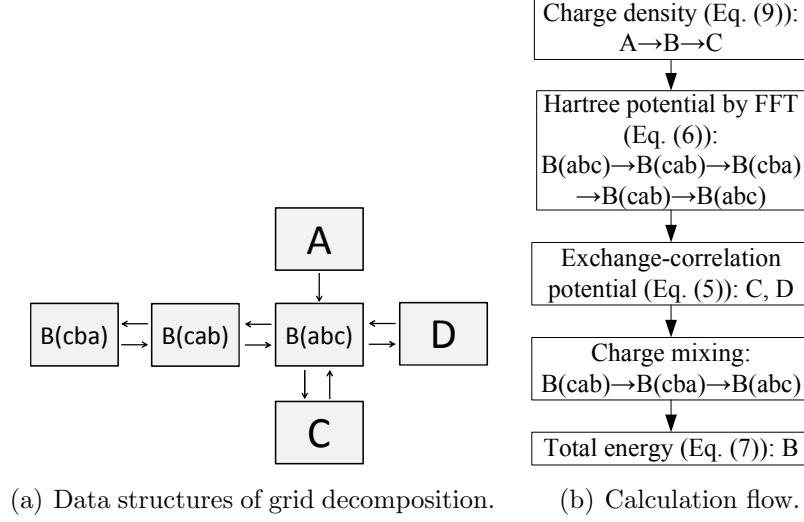


Figure 10: (a) Data structures and (b) calculation flow.

D based on their definition. The data structure for transferring $\rho_i(\mathbf{r})$ from the structure A to the structure B when the charge density $\rho(\mathbf{r})$ is calculated in the structure B using $\rho_i(\mathbf{r})$ is constructed. Similarly, the data structure for transferring $\rho(\mathbf{r})$ from the structure B to the structure C when $\rho(\mathbf{r})$ is constructed in C using $\rho(\mathbf{r})$ stored in B, and the data structure for transferring $\rho(\mathbf{r})$ from the structure B to the structure D for the case that $\rho(\mathbf{r})$ is constructed in D using $\rho(\mathbf{r})$ stored in B are constructed. In order to calculate the Hartree potential by FFT with different distributions of the structure B, the data structures for MPI communications in FFT from **abc** to **cab** and from **cab** to **cba** are also built. It is worth mentioning that we employ non-blocking MPI routines `MPI_Isend()` and `MPI_Irecv()` for point-to-point communications between pairs of processes to cut the number of communication steps between the processes and hence, reducing their waiting time. This implementation is definitely much more efficient than posting pairs of blocking `MPI_Send()` and `MPI_Recv()`.

Step 3 $O(N)$ calculation. The linear scaling Krylov subspace method is implemented in this step. Although the atoms have already been allocated nearly equally to the MPI processes in Step 1, here we further

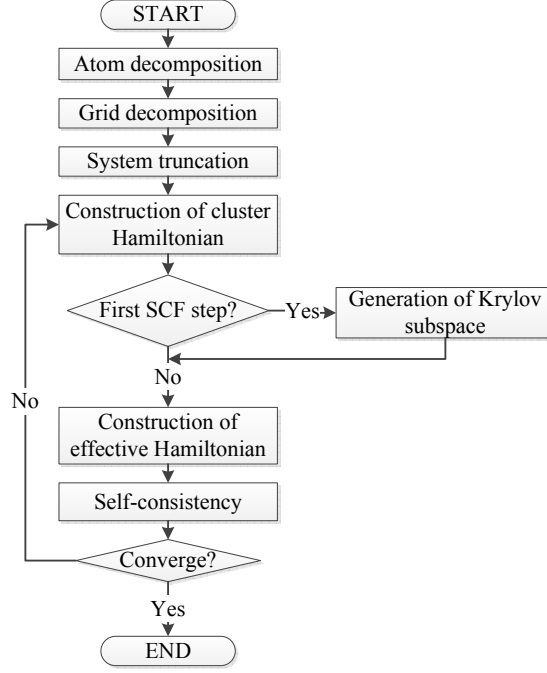


Figure 11: Computational flow.

decompose the allotted atoms to the processing cores in each MPI process using OpenMP thread parallelization in the hybrid MPI/OpenMP implementation.

Step 3a **System truncation:** Physical truncation scheme [19] is employed to divide the large system into smaller truncated clusters. It collects all the neighboring sites within a sphere with a certain cutoff radius to construct the corresponding clusters.

Step 3b **Construction of cluster Hamiltonian:** The Hamiltonian of each truncated cluster is constructed in the conventional fashion [50]. The matrix elements in each site can be calculated independently in parallel, and the cluster Hamiltonian is built by gathering all the elements from the processes through communications. Also, the Hartree potential is determined from all the contributions from the truncated clusters to ensure high accuracy.

Table 1: Software configuration.

Parameter	Setting
OpenMX	3.637 (developer release)
level.of.stdout	1
level.of.fileout	0
scf.XcType	GGA-PBE
scf.maxIter	10
scf.EigenvalueSolver	krylov (O(N) method)
scf.ProExpn.VNA	off
MD.Type	nomd

- Step 3c **Generation of Krylov subspace:** For numerical robustness, the Krylov subspace of each truncated cluster is generated only at the first SCF step, and is re-utilized in subsequent steps [34]. The Krylov subspace generation is independent and performed in parallel.
- Step 3d **Construction of effective Hamiltonian:** The effective Hamiltonian is constructed by determining the short and long-range contributions, which can be performed in parallel on each process. For truncated clusters with a large buffer region, the long-range contribution is calculated only at the first SCF step and re-used in subsequent steps.
- Step 3e **Self-consistency:** The standard eigenvalue problem with the effective Hamiltonian is solved, and the eigenvectors are obtained by performing the back transformation to calculate the charge density. After that, a common chemical potential that conserves the total number of electrons for all the truncated clusters is determined by a bisection method, where MPI communication is performed using MPI_Allreduce [34].

5. Benchmark results

5.1. Software and system configuration

5.1.1. OpenMX

All the calculations are performed using OpenMX [54], which is an open source parallel software package developed based on DFT, norm-conserving pseudopotentials, and pseudo-atomic localized basis functions. In particular, we use OpenMX version 3.6, release 37 (developer release) with a focus on

Table 2: System configuration.

Parameter	Setting
Number of cores	XT5 and FX10: Up to 2,048 K computer: Up to 131,072
Number of threads	XT5: 1 (flat MPI), and 4 and 8 (hybrid) FX10: 1 (flat MPI), and 8 and 16 (hybrid) K computer: 8 (hybrid)
Number of atoms	XT5 and FX10: Up to 65,536 K computer: Up to 262,144

the linear scaling Krylov subspace method. The number of SCF iterations is set at 10 for each calculation with no geometry optimization. In addition, GGA is employed to calculate the exchange-correlation potential, a required energy cutoff of 150 Ry is specified in the real space grid techniques for numerical integrations and solving the Poisson equation using FFT, while the cutoff energy is determined by the unit cell size to minimize the cutoff energy difference in the a -, b -, and c -axes, and the electronic temperature is set at 300 K for counting the electron number. With regard to the Krylov subspace method, the radius of a sphere centered on each atom is fixed at 6 Å, and the dimension of the Krylov subspace in each truncated cluster is set at 400. As a result, the number of atoms in the truncated cluster is 159 with the diamond structure. With these settings in our method, we confirm that the absolute error in the total energy is 0.00072 Hartree/atom in self-consistent calculations with the diamond structure, in comparison to the k -space conventional method performed with a large number of k points. Table 1 summarizes some important parameters.

5.1.2. System configuration

We perform the series of benchmark calculations on three different machines: Cray XT5, Fujitsu FX10, and the K computer. Each XT5 node is composed of two quad-core AMD Opteron 2.4GHz with 16GB memory, coupled with Cray SeaStar2+ interconnect, while each FX10 node has one 1.848GHz 16-core SPARC64 IXfx processor, 32GB memory, and Tofu interconnect. Each K computer node contains one 2.0GHz 8-core SPARC64 VIIIfx processor, 16GB memory, and the same Tofu interconnect as FX10. We use up to 2,048 cores on the first two machines and 131,072 cores on the K computer. On the other hand, as OpenMX is capable of executing in both flat MPI and hybrid MPI/OpenMP modes, several combinations of the number of processes and threads are applied to particular system architecture.

Specially, the calculations are carried out with 1 thread (flat MPI), 4 and 8 threads (hybrid) on XT5, 1, 8, and 16 threads on FX10, and 8 threads on the K computer. The system configuration is shown in Table 2.

5.1.3. Benchmark system

We mainly employ the diamond structure, which is well-suited to benchmark calculations due to its uniformity, with the basis function of C5.0- $s2p2$, where C represents the atomic symbol, 5.0 is the cutoff radius (Bohr) in the generation by the confinement method, and $s2p2$ indicates that two radial functions are used to expand the basis functions for s - and p - orbitals, respectively. The numbers of atoms used on XT5 and FX10 are 2,048, 4,096, 8,192, 16,384, 32,768, and 65,536 although we confirmed that systems of up to 131,072 atoms are runnable on XT5. For the K computer, we test with systems of up to 262,144 atoms. The number of atoms is also shown in Table 2. Furthermore, in order to perform test calculations with a structure different from the uniform diamond structure, we choose the long deoxyribonucleic acid (DNA) structure consisting of cytosines and guanines with 2,600, 13,000, and 26,000 atoms, and the basis functions of H7.0- $s2p1$, C7.0- $s2p2d1$, N7.0- $s2p2d1$, O7.0- $s2p2d1$, and P9.0- $s2p2d1$.

5.2. Results

To minimize the impact of system performance variations, each calculation is executed five times, except for the K computer, and the average values are taken as the results of the calculation. The following subsections present the calculation results, divided into a demonstration of the 3D atom decomposition method, linear scaling property by the Krylov subspace method, strong and weak scaling properties with the diamond structure, strong scaling property with the DNA structure, and strong scaling property of sub-routines, followed by an analysis on memory usage.

5.2.1. 3D atom decomposition

Figure 12 demonstrates the operation of our 3D atom decomposition in the cases of 16,384 diamond atoms and 19 MPI processes, and multiply-connected carbon nanotubes consisting of 564 carbon atoms with 8 and 16 MPI processes, where atoms of the same color are allocated to the same MPI process. In the diamond case, even though 19 is a prime number meant to pose challenges to general decomposition schemes, our method proves to be able to work efficiently, and atoms are practically localized to the process

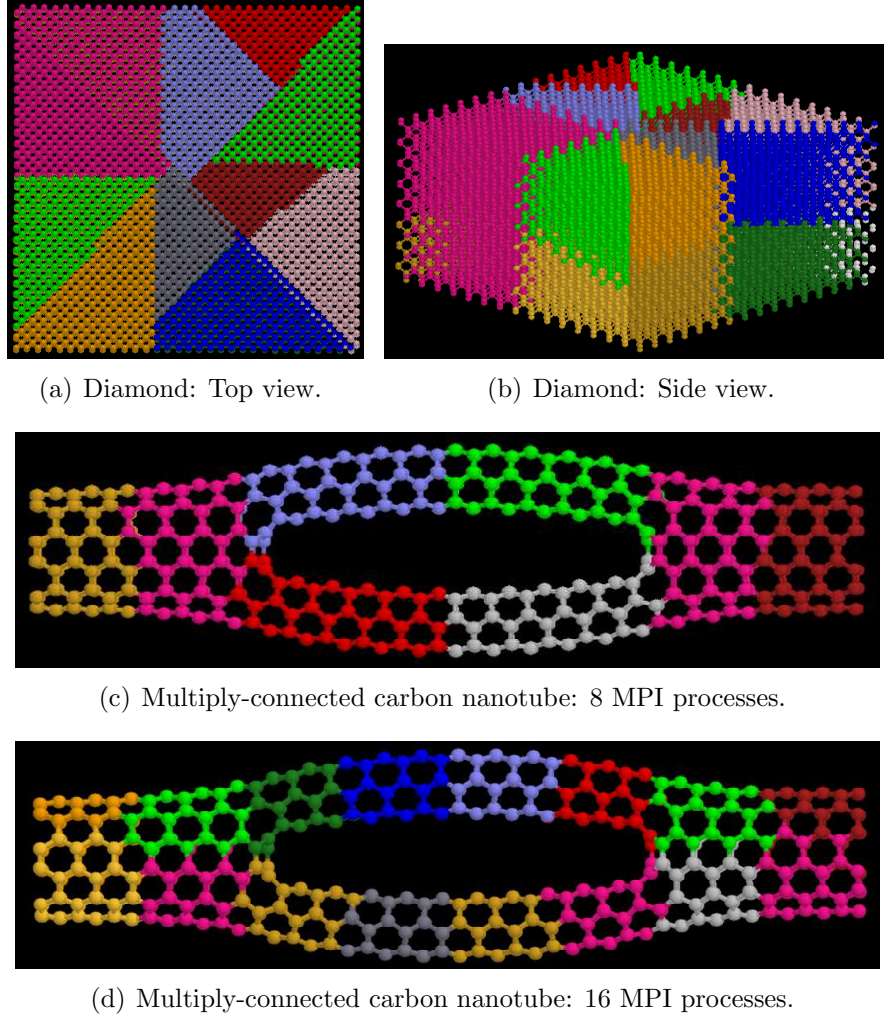
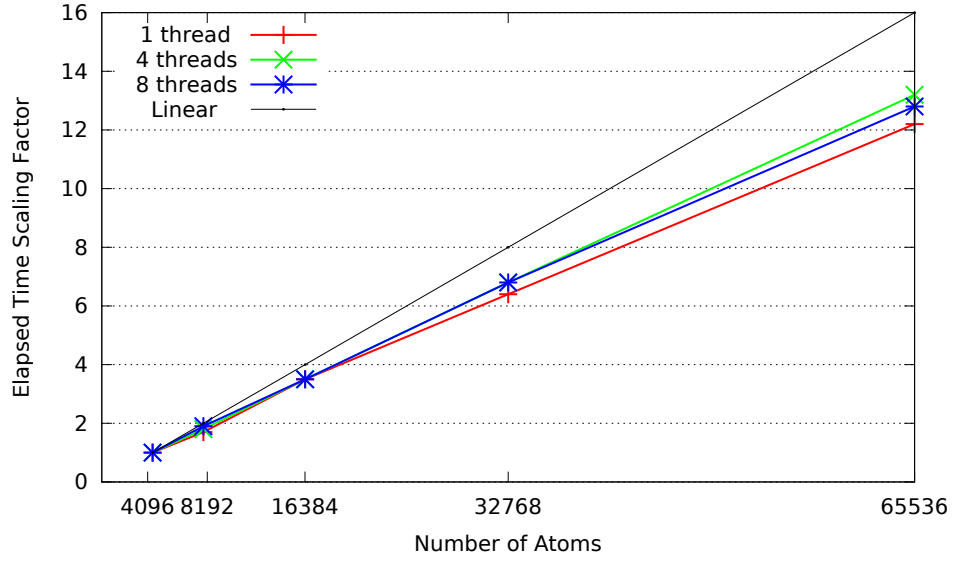


Figure 12: 3D atom decomposition examples: 16,384 diamond atoms and 19 MPI processes with (a) top view and (b) side view, and multiply-connected carbon nanotubes consisting of 564 carbon atoms with (c) 8 MPI processes and (d) 16 MPI processes. Atoms of the same color are allocated to the same MPI process.

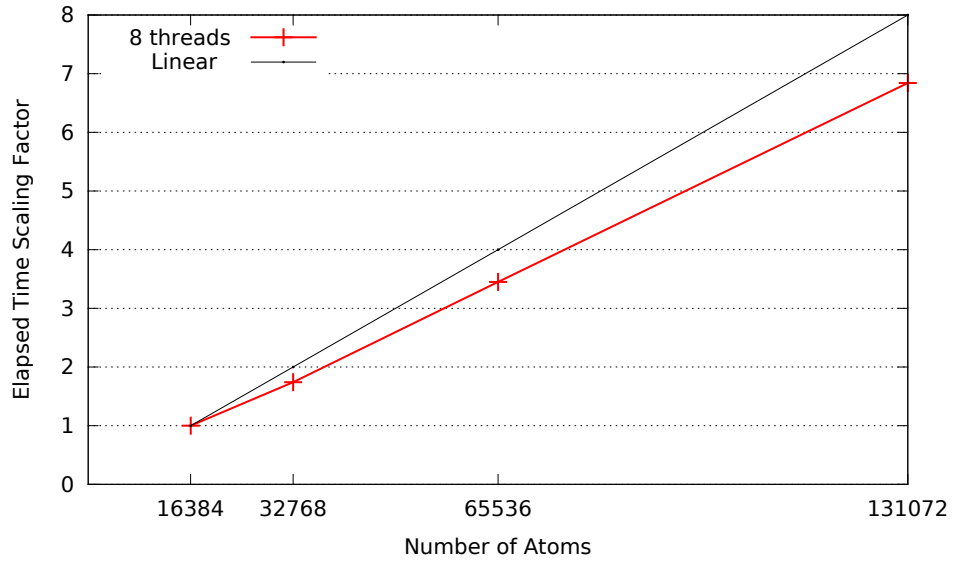
holding them. Similar atom locality conservation is also obtained with the multiply-connected carbon nanotubes, where nearby atoms are distributed to the same process. This is of great importance to reduce communication amount and memory usage for performance enhancement. Although our examples are limited to up to 19 processes, it is guaranteed that the systems are well divided for many processes, since the decomposition is recursively applied to sub-systems.

5.2.2. *Linear scaling property*

With the $O(N)$ Krylov subspace method, the elapsed time should be approximately linear to the number of atoms when the number of cores is fixed. Figure 13 presents the relationship between the elapsed time scaling factor and the number of atoms with a fixed number of cores of 2,048 on XT5 (a), and 16,384 cores on the K computer (b) with the diamond structure. The relationship appears to be nearly linear: an increase in the number of atoms by a factor of 2 leads to an increase in the elapsed time by a factor of a little bit less than 2 in all three cases of 1, 4, and 8 threads. In case of 65,536 atoms in Fig. 13(a), the largest and smallest elapsed time scaling factors are 13.2 (4 threads) and 12.1 (1 thread), respectively, as opposed to the linear factor of 16 against the baseline of 4,096 atoms. The same behavior can be observed on the K computer, as shown in Fig. 13(b). The reason behind the better-than-linear elapsed time scaling factors is a higher computation to communication ratio with a higher number of atoms per core. When the number of cores is fixed and the number of atoms is increased, the computational amount per core will also be increased, leading to a higher efficiency. For instance, with a fixed number of 16,384 cores on the K computer (Fig. 13b), the number of atoms per core in case of 16,384 atoms is only 1, while this number is 8 for 131,072 atoms, resulting in the computational amount per core with 131,072 atoms being 8 times larger than that with 16,384 atoms. Suppose the communication amount stays almost unchanged for a fixed number of cores, the computation to communication ratio with 131,072 atoms would also be 8 times higher than that with 16,384 atoms. Such a linear scaling property is definitely essential to enable extreme-scale applications that require hundreds of thousands to millions of atoms. The property will also be demonstrated with the DNA structure in 5.2.4.

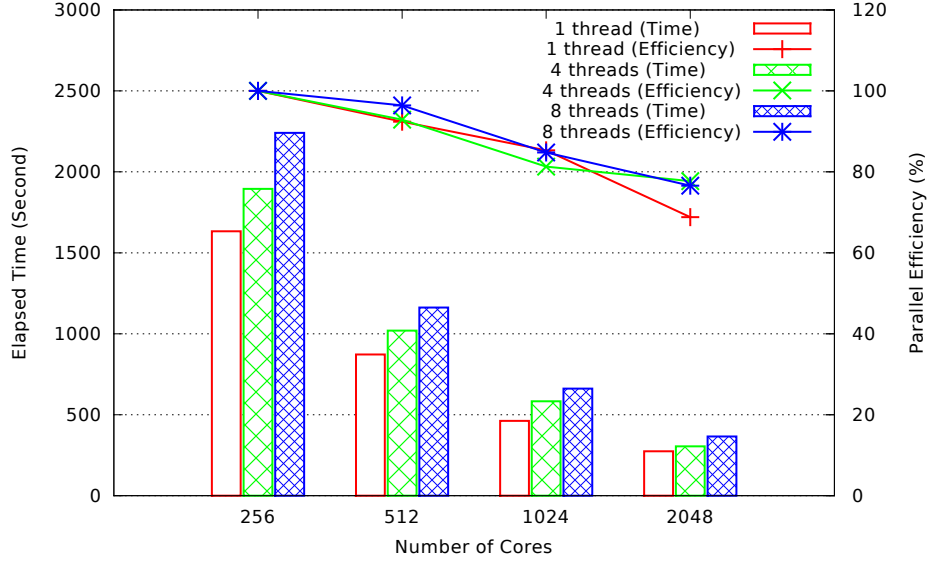


(a) XT5. The number of cores is fixed at 2,048.

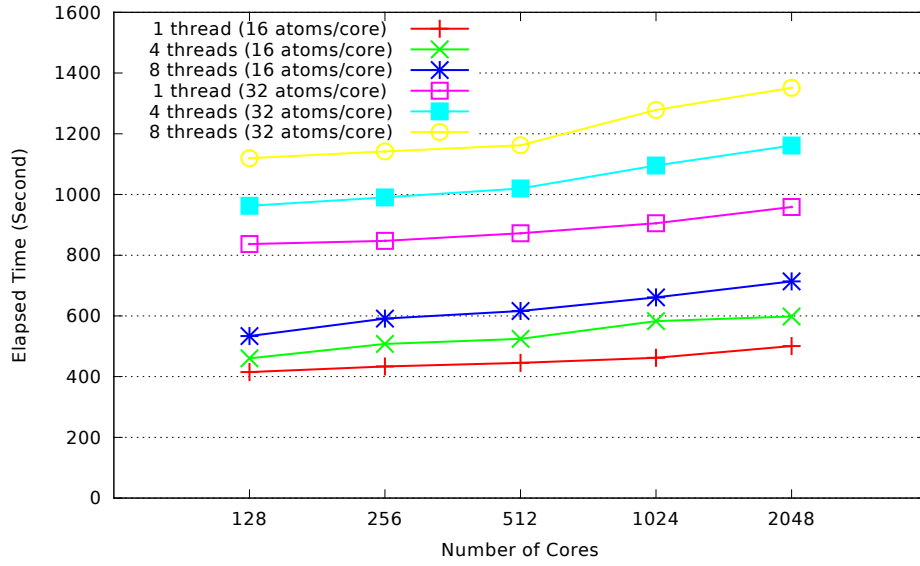


(b) K computer. The number of cores is fixed at 16,384.

Figure 13: Linear scaling property with the diamond structure.

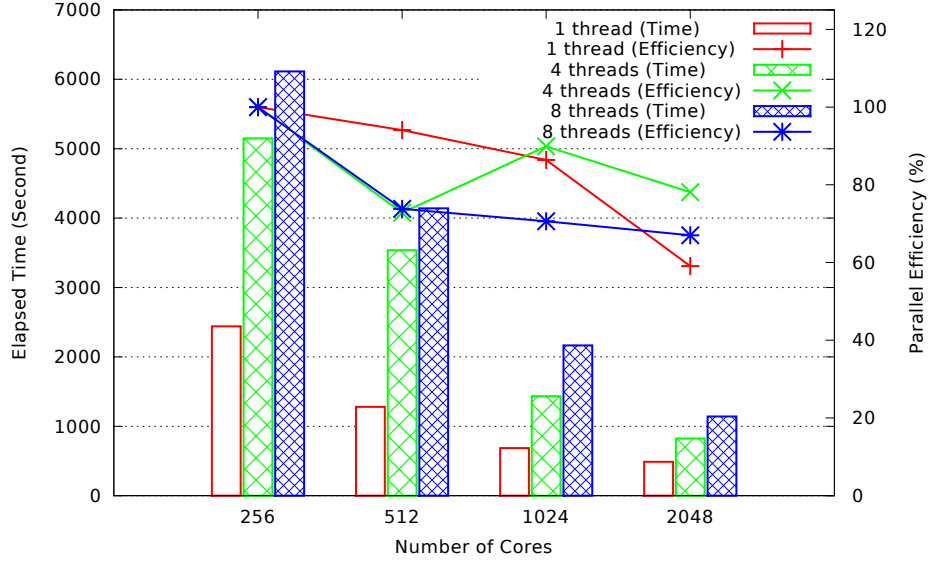


(a) Strong scaling with 16,384 atoms.

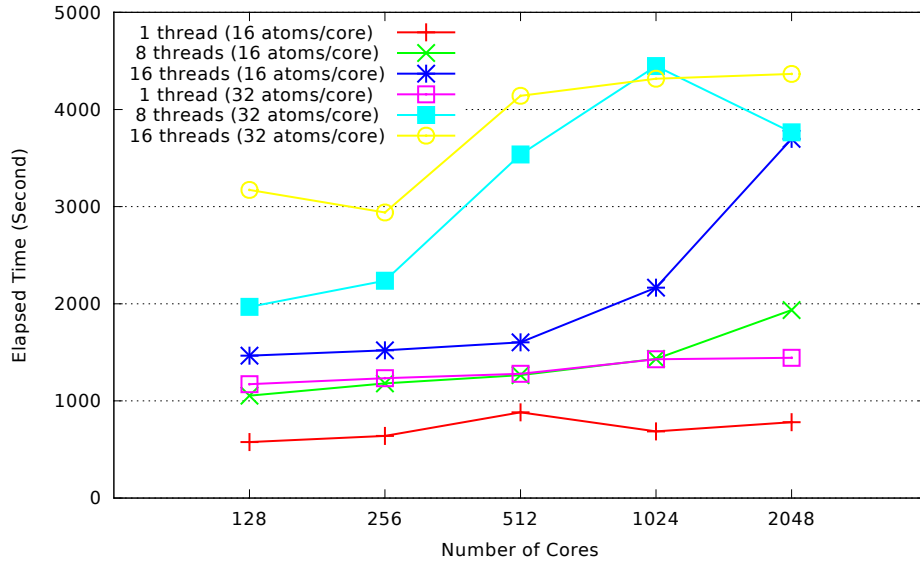


(b) Weak scaling with 16 and 32 atoms per core.

Figure 14: (a) Strong and (b) weak scaling on XT5 with the diamond structure.

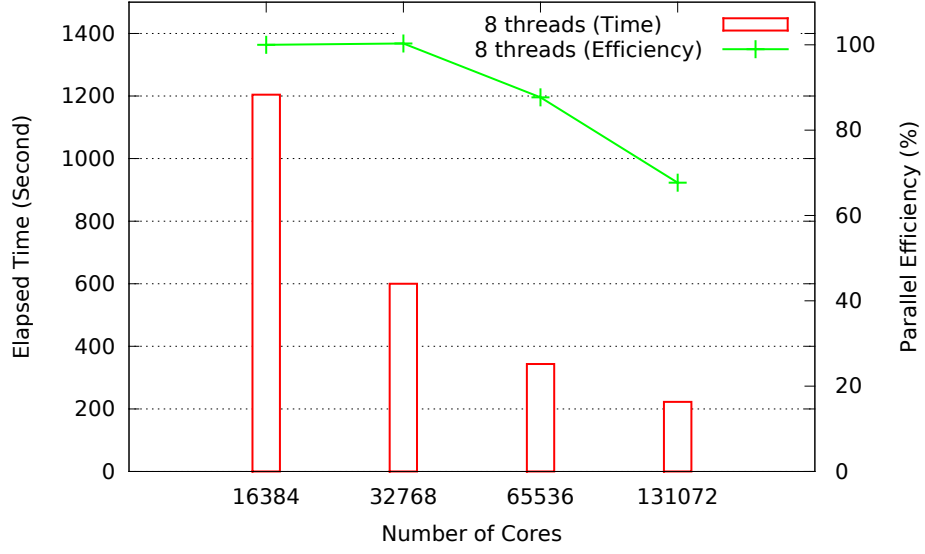


(a) Strong scaling with 16,384 atoms.

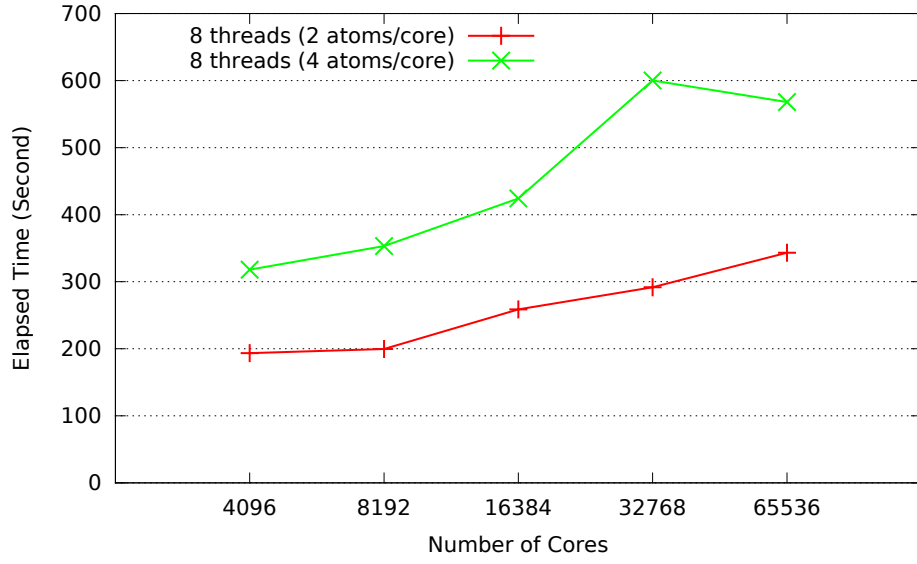


(b) Weak scaling with 16 and 32 atoms per core.

Figure 15: (a) Strong and (b) weak scaling on FX10 with the diamond structure.



(a) Strong scaling with 131,072 atoms.



(b) Weak scaling with 2 and 4 atoms per core.

Figure 16: (a) Strong and (b) weak scaling on the K computer with the diamond structure.

5.2.3. Strong and weak scaling with the diamond structure

Figures 14, 15 and 16, respectively, show the strong scaling property in terms of parallel efficiency and elapsed time, and weak scaling property in terms of elapsed time on XT5, FX10, and the K computer with the diamond structure. In strong scaling calculations, the number of atoms allocated to one processing core is decreased with the increasing number of cores, resulting in a reduction in elapsed time. The strong calculations are performed with 16,384 atoms on both XT5 and FX10 from 256 to 2,048 cores, each with 1, 4, and 8 threads on XT5, and with 1, 8, and 16 threads on FX10. On the K computer, the strong scaling results are taken with 131,072 atoms and up to 131,072 cores in the hybrid mode (8 threads). The parallel efficiency of strong scaling is given by the following equation:

$$P_C = \frac{\frac{T_B}{T_C}}{\frac{N_C}{N_B}} \times 100\%, \quad (20)$$

where P_C is the parallel efficiency of the target calculation, T_B and N_B are the elapsed time and number of cores of the base calculation, respectively, and T_C and N_C are the elapsed time and number of cores of the target calculation, respectively. The base calculation used the equation is the case of 256 cores with XT5 and FX10, and the case of 16,384 cores with the K computer, the lowest numbers of cores.

On the other hand, weak scaling calculations demand the number of atoms allocated to one processing core be fixed when changing the numbers of cores and atoms to maintain a constant elapsed time in the ideal case, due to a constant computational amount per core. The number of atoms per core in the weak calculations is set at 16 and 32 from 128 cores to 2,048 cores on XT5 and FX10, and 2 and 4 from 4,096 cores to 65,536 cores on the K computer. For example, on XT5 and FX10, in case of 16 atoms per core and 128 cores, there are 2,048 atoms to be processed, and in another case of 2,048 cores and 32 atoms per core, we have to calculate 65,536 atoms.

We note several observations. First of all, both flat MPI and hybrid MPI/OpenMP modes exhibit good strong scaling property, with the parallel efficiency ranging from 68.8% to 96.4% on XT5 (Fig. 14(a)), and from 59.1% to 94.1% on FX10 (Fig. 15(a)). On the K computer, the parallel efficiency at 131,072 cores is 67.7% compared to the baseline of 16,384 cores (Fig. 16(a)). Also, the parallel efficiency of the hybrid mode tends to be higher than that of the flat MPI mode, especially with the largest number of cores, probably

due to a smaller number of MPI processes, accompanied by a smaller amount of communication, for the same number of cores. Using 2,048 cores in the flat MPI mode, the calculation of 16,384 atoms could be finished in less than 250 seconds on XT5.

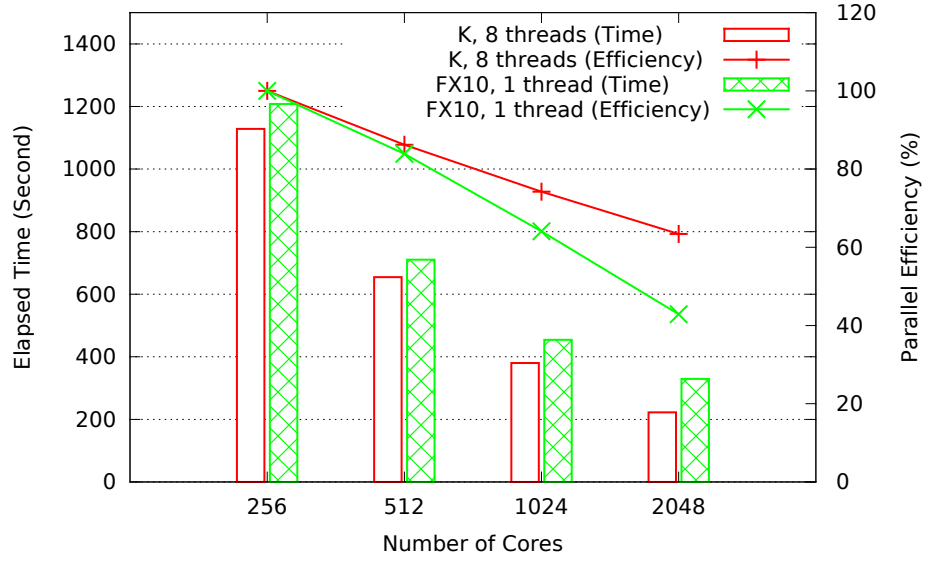
For weak scaling on XT5 (Fig. 14(b)), the elapsed time increases gradually with the number of cores due to a growing communication amount caused by more MPI processes. Nevertheless, the increase rate is quite small, ranging from 14.6% (1 thread with 32 atoms/core) to 33.8% (8 threads with 16 atoms/core), when the number of cores is raised by a factor of 16 from 128 to 2,048. More importantly, the efficiency maintains its property when the calculation scale is enlarged gradually from 2,048 to 32,768 atoms (16 atoms/core), and from 4,096 to 65,536 atoms (32 atoms/core). This observation suggests that we can expect similar performance for even larger scales, as shown on the K computer (Fig. 16(b)). On FX10 (Fig. 15(b)), unfortunately we experience some sudden jumps in elapsed time with the hybrid mode, while the flat MPI mode shows a very similar scaling property to that on XT5 with a solid performance (in the same Fig. 15(b)).

In comparison between the flat MPI and hybrid modes, the elapsed time of the flat MPI mode is always lower than that of the hybrid mode with 4 threads, which in turn is always lower than that of the hybrid mode with 8 threads. The outcome is the same for both strong and weak scaling, asserting that flat MPI is the best mode in both machines at this calculation scale.

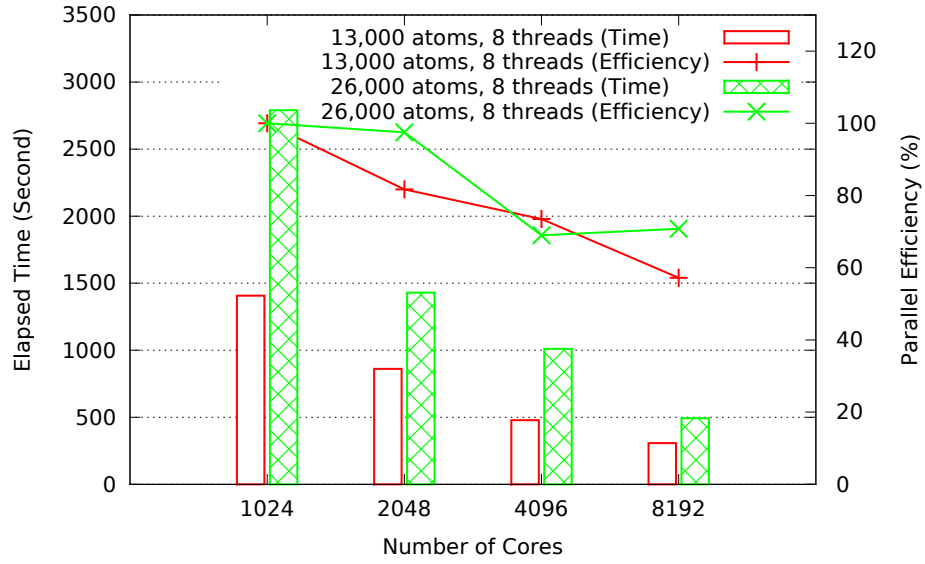
Finally, in weak scaling on XT5 with both hybrid and flat MPI modes and on FX10 with the flat MPI mode, when the number of atoms per core is increased by a factor of 2 from 16 to 32, the elapsed time also increases almost exactly by a factor of 2 as expected. The results again confirm the linear scaling property of OpenMX, and that it can maintain the same level of efficiency for larger calculation scales, as demonstrated on the K computer.

5.2.4. Strong scaling property with DNA

In the diamond case, the structure is uniform, and the numbers of atoms and cores are selected so that each core is assigned the same computational amount. In the case of DNA, by contrast, the structure is long, and the number of atoms is not divisible by the number of cores. Even so, our decomposition scheme is still able to work effectively for the DNA structure, as shown in Fig. 17, where the parallel efficiency is calculated by Eq. (20) with the baseline being 256 cores in Fig. 17(a) and 1,024 cores in Fig. 17(b). With 2,600 atoms and up to 2,048 cores on the K computer (hybrid with 8



(a) DNA with 2,600 atoms on the K computer and FX10.



(b) DNA with 13,000 and 26,000 atoms on the K computer.

Figure 17: Strong scaling property with the DNA structure.

threads) and FX10 (flat MPI) (Fig. 17(a)), the parallel efficiency at 2,048 cores is 63.4% in the hybrid mode, and drops to only 42.9% in the flat MPI mode. The efficiency gap is due to a load imbalance caused by the difference in the number of processes in these modes. In the hybrid mode, the number of processes at 2,048 cores is only 256, as against 2,048 in the flat MPI mode. Assume that the atoms have the same weight of 1. Because there are 2,600 atoms, each process is allocated 10 or 11 atoms in the hybrid mode, and 1 or 2 atoms in the flat MPI mode. Obviously, the load imbalance of the flat MPI mode is 100%, much higher than 10% of the hybrid mode, resulting in the low efficiency.

Figure 17(b) compares the elapsed time and parallel efficiency of 13,000 and 26,000 atoms in the hybrid mode on the K computer with up to 8,192 cores. By observing the elapsed time at each number of cores, we can easily recognize the better-than-linear behavior of our Krylov subspace method, where the elapsed time of 26,000 atoms is nearly twice as long as that of 13,000 atoms. In addition, the parallel efficiency of 26,000 atoms is much higher than that of 13,000 atoms, except for 4,096 cores, for example, 70.8% compared to 57.2% at 8,192 cores. The result is attributed to the computational amount per core that is doubled in case of 26,000 atoms, leading to a higher computation to communication ratio and eventually a higher efficiency.

5.2.5. Strong scaling property of subroutines

So far we have discussed only the total elapsed time, which should be broken down further to understand the scaling behavior of the subroutines in DFT. Figure 18 plots the strong scaling property of the subroutines, taken with 16,384 diamond atoms in the flat MPI mode on XT5. Some important and time-consuming subroutines are noted below in order of descending elapsed time.

- `Diagonalization()`: is a subroutine for performing the diagonalization, which is actually the $O(N)$ Krylov subspace method in the benchmark series. It accounts for 48.7% of the total time, and scales well, reaching a speedup of 7.3 at 2,048 cores compared to the ideal speedup of 8.
- `Set_Nonlocal()`: is a subroutine for calculating the matrix elements and its derivatives for nonlocal potentials in the momentum space, accounting for 19.6% of the total time. The speedup at 2,048 cores is 6.5, also

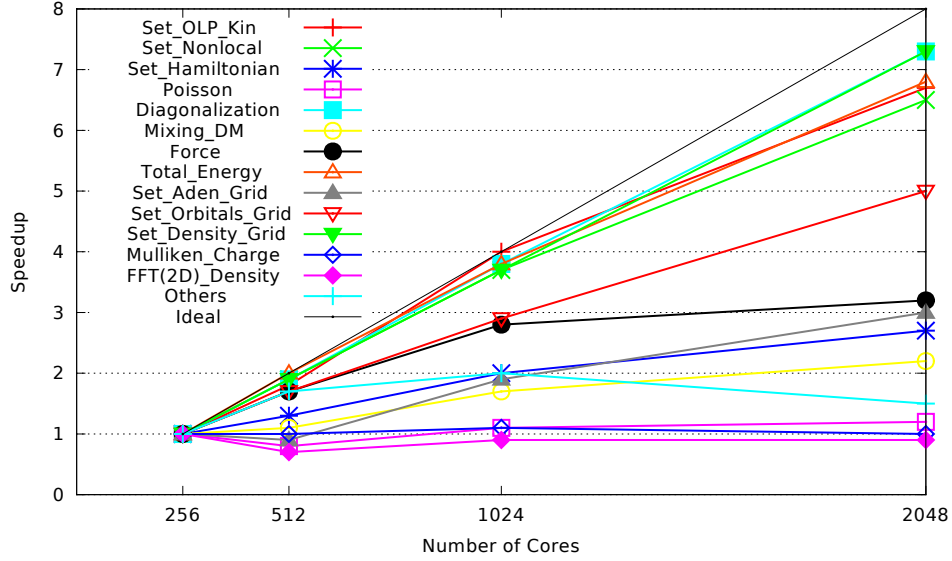


Figure 18: Strong scaling property of subroutines. The results are taken with 16,384 diamond atoms, flat MPI on XT5.

good considering that fact that MPI communication is incurred in this subroutine.

- `Force()`: As the name reveals, this subroutine calculates the force on atoms and contributes 9.4% to the total time. This is the worst scaling subroutine among the time-consuming ones, as the speedup is only 3.2, less than half of the ideal speedup of 8, caused by a large amount of MPI communication among the processes. This amount can be dramatically reduced if buffer regions are added to store more data, with the tradeoff of incurring high memory usage. Nevertheless, the subroutine is called only once in each MD step, and hence, becoming smaller in calculations that require a large number of SCF iterations. We are now in the process of tuning this subroutine to make it scale better.
- `Total_Energy()`: is a subroutine for calculating the total system energy, accounting for 8.3% of the total time. It exhibits a good scaling property with the speedup of 6.8 at 2,048 cores.
- `Set_OLP_Kin()`: is a subroutine for calculating the overlap matrix and the matrix for the kinetic operator in the momentum space. It accounts

for 6.5% of the total time and also scales well with a speedup of 6.7.

- `Poisson()`: is a subroutine for solving the Poisson's equation using FFT. Although its contribution to the total time is trivial, approximately 0.02%, we note it here as it is directly related to our method for grid decomposition.

In summary, all the time-consuming subroutines, except for `Force()`, which contribute more than 5% to the total time, scale well with the speedup of 6.5 to 7.3, as against the ideal speedup of 8. As a result, the total time shows a good scaling property as can be seen in previous subsections.

5.2.6. Memory usage

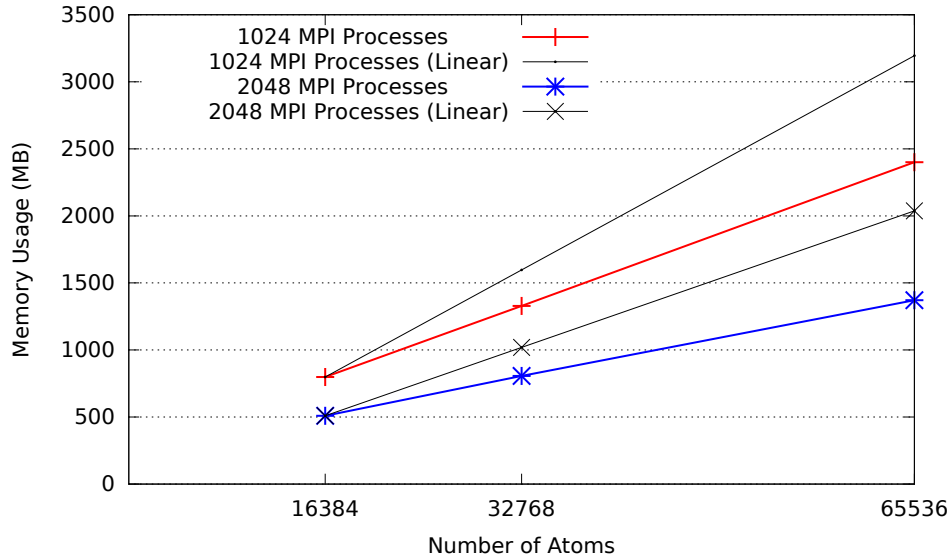


Figure 19: Memory usage per process with the diamond structure, measured with 1,024 and 2,048 MPI processes in flat MPI mode on XT5.

Memory management is another crucial aspect that must be handled properly in large-scale calculations. As touched on above, we give priority to efficient use of memory and therefore let the processes carry out on-the-fly communications instead of extending the buffer to store more data in our implementation. Figure 19 displays the amount of memory per process used in the calculations for 16,384, 32,768, and 65,536 diamond atoms, measured

with 1,024 and 2,048 MPI processes in the flat MPI mode on XT5. The numbers of grids of **a**-, **b**-, and **c**-axes are 420, 420, and 210 for 16,384 atoms, 420, 420, and 420 for 32,768 atoms, and 840, 420, and 420 for 65,536 atoms. With the number of processes fixed at 1,024 or 2,048 and the number of atoms increased linearly, the scaling property of memory usage is well below linear, with the scaling factors at 65,536 atoms are 2.7 (1,024 processes) and 3.0 (2,048 processes), as against the linear factor of 4. On the other hand, when the number of processes is doubled from 1,024 to 2,048, the memory usage efficiency factors are 78.4%, 82.4%, and 87.6% for 16,384, 32,768, and 65,536 atoms, respectively.

6. Conclusion

We have developed a three-dimensional domain decomposition scheme that is applicable to both conventional DFT methods and linear scaling DFT methods, consisting of two methods: one for atom decomposition and the other for grid decomposition. The atom decomposition method reorders the atoms along a principal axis based on a modified recursive bisection method and inertia tensor moment. The atoms that are close in real space are also close on the principal axis to maintain data locality. The atoms are then divided into sub-domains in a balanced way among the processes. On the other hand, the grid decomposition method makes it easier for the calculations of charge density and other potentials by defining different data structures for storing the grid data. Also, our proposed decomposition method for solving the Poisson equation using FFT in the calculation of Hartree potential exhibits up to 77.8% enhancement of communication efficiency in comparison to a previously proposed method with $64 \times 64 \times 64$ grid points. Our scheme has been evaluated with OpenMX and the linear scaling Krylov subspace method, and demonstrates good strong and weak scaling properties. On the K computer, the parallel efficiency at 131,072 cores is 67.7% compared to the baseline of 16,384 cores with 131,072 diamond atoms. The efficiency is from 68.8% to 96.4% on XT5, and from 59.1% to 94.1% on FX10 with up to 2,048 cores, depending on the number of cores in use. The results suggest that our scheme is efficient for enabling large-scale electronic calculations based on DFT on massively parallel computers.

We are now in the process of tuning the performance of some particular subroutines, especially Force(), the worst scaling subroutine among the time-consuming ones. Should it be improved, certainly we could further accelerate

the calculations and achieve a higher parallel efficiency.

Acknowledgements

This work was supported by the Strategic Programs for Innovative Research (SPIRE), MEXT, and the Computational Materials Science Initiative (CMSI), and Materials Design through Computics: Complex Correlation and Non-Equilibrium Dynamics A Grant in Aid for Scientific Research on Innovative Areas, MEXT, Japan. The benchmark calculations were performed using the K computer at RIKEN, Fujitsu FX10 at The University of Tokyo, and Cray XT5 at Japan Advanced Institute of Science and Technology (JAIST).

References

- [1] P. Hohenberg, W. Kohn, Phys. Rev. 136 (1964) B864–B871.
- [2] W. Kohn, L. J. Sham, Phys. Rev. 140 (1965) A1133–A1138.
- [3] L. Hedin, Phys. Rev. 139 (1965) A796–A823.
- [4] Y. Hasegawa, J.-I. Iwata, M. Tsuji, D. Takahashi, A. Oshiyama, K. Minami, T. Boku, F. Shoji, A. Uno, M. Kurokawa, H. Inoue, I. Miyoshi, M. Yokokawa, in: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC '11, ACM, New York, NY, USA, 2011, pp. 1:1–1:11.
- [5] Y. Saad, J. Chelikowsky, S. Shontz, SIAM Rev. 52 (2010) 1.
- [6] E. Tsuchida, M. Tsukada, Phys. Rev. B 52 (1995) 5573–5578.
- [7] E. Tsuchida, M. Tsukada, Phys. Rev. B 54 (1996) 7602–7605.
- [8] E. Tsuchida, M. Tsukada, J. Phys. Soc. Jpn. 67 (1998) 3844–3858.
- [9] J. Pask, P. Sterne, Model. Simul. Mater. Sci. Eng. 13 (2005) R71.
- [10] S. Goedecker, Rev. Mod. Phys. 71 (1999) 1085.
- [11] D. Bowler, T. Miyazaki, M. Gillan, J. Phys.: Condens. Matter 14 (2002) 2781.
- [12] S. Goedecker, G. Scuserza, Comp. Sci. Eng. 5 (2003) 14–21.

- [13] D. Bowler, T. Miyazaki, Rep. Prog. Phys. 75 (2012) 036503.
- [14] C. Skylaris, P. Haynes, A. Mostofi, M. Payne, J. Chem. Phys. 122 (2005) 084119.
- [15] W. Yang, Phys. Rev. Lett. 66 (1991) 1438–1441.
- [16] W. Yang, T. Lee, J. Chem. Phys. 103 (1995) 5674.
- [17] J. Khandogin, K. Musier-Forsyth, D. York, J. Mol. Biol. 330 (2003) 993–1004.
- [18] T. Ozaki, Phys. Rev. B 59 (1999) 16061–16064.
- [19] T. Ozaki, M. Aoki, D. G. Pettifor, Phys. Rev. B 61 (2000) 7972–7988.
- [20] T. Ozaki, K. Terakura, Phys. Rev. B 64 (2001) 195126.
- [21] W. Kohn, Phys. Rev. Lett. 76 (1996) 3168–3171.
- [22] U. Stephan, D. Drabold, Phys. Rev. B 57 (1998) 6391.
- [23] P. Ordejón, D. Drabold, R. Martin, M. Grumbach, Phys. Rev. B 51 (1995) 1456.
- [24] F. Mauri, G. Galli, R. Car, Phys. Rev. B 47 (1993) 9973.
- [25] H. Xiang, Z. Li, W. Liang, J. Yang, J. Hou, Q. Zhu, J. Chem. Phys. 124 (2006) 234108.
- [26] E. Rudberg, E. Rubensson, J. Phys.: Condens. Matter 23 (2011) 075502.
- [27] D. Jordan, D. Mazziotti, J. Chem. Phys. 122 (2005) 084114.
- [28] M. S. Daw, Phys. Rev. B 47 (1993) 10895–10898.
- [29] X.-P. Li, R. W. Nunes, D. Vanderbilt, Phys. Rev. B 47 (1993) 10891–10894.
- [30] S. Goedecker, L. Colombo, Phys. Rev. Lett. 73 (1994) 122–125.
- [31] S. Goedecker, M. Teter, Phys. Rev. B 51 (1995) 9455–9464.
- [32] F. R. Krajewski, M. Parrinello, Phys. Rev. B 71 (2005) 233105.

- [33] F. R. Krajewski, M. Parrinello, Phys. Rev. B 73 (2006) 041105.
- [34] T. Ozaki, Phys. Rev. B 74 (2006) 245101.
- [35] D. Bowler, T. Miyazaki, J. Phys.: Condens. Matter 22 (2010) 074207.
- [36] E. Artacho, E. Anglada, O. Diéguez, J. Gale, A. García, J. Junquera, R. Martin, P. Ordejón, J. Pruneda, D. Sánchez-Portal, et al., J. Phys.: Condens. Matter 20 (2008) 064208.
- [37] P. Haynes, C. Skylaris, A. Mostofi, M. Payne, Phys. Stat. Sol. B 243 (2006) 2489–2499.
- [38] V. Brázdová, D. Bowler, J. Phys.: Condens. Matter 20 (2008) 275223.
- [39] M. Challacombe, Comp. Phys. Comm. 128 (2000) 93–107.
- [40] W. Hehre, L. Radom, P. Schleyer, J. Pople, et al., Ab initio molecular orbital theory, volume 9, Wiley, New York, 1986.
- [41] D. Sánchez-Portal, P. Ordejon, E. Artacho, J. Soler, Int. J. Quant. Chem. 65 (1997) 453–461.
- [42] T. Ozaki, Phys. Rev. B 67 (2003) 155108.
- [43] L. Thomas, in: Proc. Camb. Phil. Soc., volume 23, Cambridge Univ Press, pp. 542–548.
- [44] J. Slater, Phys. Rev. 81 (1951) 385.
- [45] R. Parr, W. Yang, Density-functional theory of atoms and molecules, volume 16, Oxford Univ. Pr., 1994.
- [46] R. Martin, Electronic structure: basic theory and practical methods, Cambridge Univ. Pr., 2004.
- [47] T. Ohwaki, M. Otani, T. Ikeshoji, T. Ozaki, J. Chem. Phys. 136 (2012) 134101.
- [48] H. Sawada, S. Taniguchi, K. Kawakami, T. Ozaki, submitted (2012).
- [49] J. Salmon, Parallel hierarchical N-body methods, Physics, Mathematics and Astronomy Pub., California Institute of Technology, 1991.

- [50] T. Ozaki, H. Kino, Phys. Rev. B 72 (2005) 045121.
- [51] J. Perdew, K. Burke, M. Ernzerhof, Phys. Rev. Lett. 77 (1996) 3865–3868.
- [52] J. Perdew, A. Zunger, Phys. Rev. B 23 (1981) 5048.
- [53] D. Takahashi, in: R. Wyrzykowski, J. Dongarra, K. Karczewski, J. Wasniewski (Eds.), Parallel Processing and Applied Mathematics, volume 6067 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2010, pp. 606–614.
- [54] OpenMX, Open source package for Material eXplorer, <http://www.openmx-square.org/>, retrieved 2012-08-20.